AMENDMENTS TO THE CLAIMS:

This listing of claims will replace all prior versions and listings of claims in the application:

1. (Canceled)

2. (Canceled)

3. (Currently amended) A method performed on a processor operatively coupled to a collection of servers, which enables a client associated with the processor to dynamically distribute a task to a server, the method comprising ~~the steps of~~:

selecting a server to process the task;

forming a task request from parameters and data;

sending the task request to the selected server ~~which downloads any needed executable byte code~~, wherein the selected server:

downloads a class definition after receiving the task request, wherein the class definition maps locations of information in the task request and allows the selected server to process the task request;

extracts parameters and data from the task request using the downloaded class definition; and

invokes a generic compute technique capable of executing a plurality of types of tasks, wherein the generic compute technique executes the task request ~~on the selected server~~ using the extracted parameters and data ~~and generates results~~; and

receiving ~~the~~ results associated with the executed task request ~~back~~ from the selected server.

4. (Original) The method of claim 3, wherein the processor is operatively coupled to a computer system having a primary storage device, a secondary storage device, a display device, and an input/output mechanism.

5. (Original) The method of claim 3, wherein the task is developed in a programming language and environment compatible with each of the server computers.

6. (Original) The method of claim 3, wherein the server is selected from a plurality of heterogeneous computer systems.

7. (Original) The method of claim 5, wherein the environment includes a remote procedure call subsystem.

8. (Original) The method of claim 7, wherein the remote procedure call subsystem is the Remote Method Invocation (RMI) system.

9. (Previously presented) The method of claim 3, wherein selecting the server comprises selecting the server based on the overall processing load distribution among the collection of servers.

10. (Original) The method of claim 6, wherein the selected server has the lowest load characteristic compared with average load characteristic of the servers over a predetermined time period.

11. (Previously presented) The method of claim 3, wherein selecting the server comprises selecting the server based on the specialized computing capabilities of each server.

12. (Previously presented) The method of claim 11, wherein the specialized computing capabilities include a capability to render images.

13. (Original) The method of claim 3, wherein the sending step further comprises the substeps of:

determining if code related to the requested task is present on the selected server; and

downloading the code onto the selected server when the code is not present on the selected server.

14. (Previously presented) The method of claim 3, wherein the sending step further comprises: providing the task as a parameter to the generic compute method.

15. (Original) The method of claim 3 further comprising the step of indicating to the server that results from a computed task should be stored in a result cache on the selected server for subsequent tasks to use.

16. (Original) The method of claim 3, wherein the results are used for further processing on the client.

17. (Previously presented) The method of claim 3, wherein the results comprise an object.

18. (Currently amended) A method performed on a processor operatively coupled to a collection of servers, which enables a server associated with the processor to dynamically receive and process a task from a client computer wherein the task is in an executable programming language compatible with each of the server computers, the method comprising ~~the steps of~~:

downloading a class definition after receiving a task request, wherein the class definition maps locations of information in the task request and allows the server to process the task request;

assembling parameters and data from ~~a~~ the task request into a task, using the downloaded class definition;

~~downloading any needed executable byte code;~~

invoking a generic compute method ~~on the server~~, ~~which is~~ capable of processing a plurality of types of tasks, on the server, wherein the generic compute method ~~which~~ executes the task and generates results; and

returning results to the client.

19. (Original) The method of claim 18, wherein the processor is operatively coupled to a computer system having a primary storage device, a secondary storage device, a display device, and an input/output mechanism.

20. (Original) The method of claim 18, wherein the task is developed in a programming language and environment compatible with each of the server computers.

21. (Original) The method of claim 18, wherein the task is developed using the Java programming language and environment.

22. (Original) The method of claim 21, wherein the environment includes a remote procedure call subsystem.

23. (Original) The method of claim 22, wherein the remote procedure call subsystem is the Remote Method Invocation (RMI) system.

24. (Previously presented) The method of claim 18, wherein the assembling step further comprises:

determining if types related to the task are available on the server;

when types are not available on the server, downloading the types onto the server from a location as indicated by the parameters provided by the client; and

executing the task based upon the data and parameters provided by the client.

25. (Original) The method of claim 24, wherein the determining step and the downloading steps are performed by a remote procedure call (RPC) subsystem.

26. (Original) The method of claim 25, wherein the determining step is performed by a Remote Method Invocation (RMI) type of remote procedure call subsystem.

27. (Original) The method of claim 18, further comprising the substep of storing the results from the task in a cache if a subsequent task will use the results.

28. (Currently amended) A computer readable medium containing instructions for controlling a computer system comprising a collection of servers to perform a method for enabling a client to dynamically distribute a task to a server, the method comprising the steps of:

selecting a server to process the task;

forming a task request from parameters and data;

sending the task request to the selected server ~~which downloads any needed executable byte code~~, wherein the selected server:

downloads a class definition after receiving the task request, wherein the class definition maps locations of information in the task request and allows the selected server to process the task request;

extracts parameters and data from the task request using the downloaded class definition; and

invokes a generic compute method capable of executing a plurality of types of tasks, wherein the generic compute technique executes the task request on the selected server using the extracted parameters and data ~~and generates results~~; and

receiving ~~the~~ results associated with the executed task request ~~back~~ from the selected server.

29. (Previously presented) The computer readable medium of claim 28, wherein the computer system is operatively coupled to a primary storage device, a secondary storage device, a display device, and an input/output mechanism.

30. (Previously presented) The computer readable medium of claim 28, wherein the task is developed in a programming language and environment compatible with each of the servers.

31. (Previously presented) The computer readable medium of claim 28, wherein the selected server is selected from a plurality of heterogeneous computer systems.

32. (Previously presented) The computer readable medium of claim 30, wherein the environment includes a remote procedure call subsystem.

33. (Previously presented) The computer readable medium of claim 32, wherein the remote procedure call subsystem is the Remote Method Invocation (RMI) system.

34. (Previously presented) The computer readable medium of claim 28, wherein selecting the server comprises selecting the server based on the overall processing load distribution among the collection of servers.

35. (Previously presented) The computer readable medium of claim 28, wherein selecting the server comprises selecting the server based on a lowest load characteristic compared to an average load characteristic of the servers over a predetermined period of time.

36. (Previously presented) The computer readable medium of claim 28, wherein selecting the server comprises selecting the server based on the specialized computing capabilities of each server.

37. (Previously presented) The computer readable medium of claim 36, wherein the specialized computing capabilities include a capability to render images.

38. (Previously presented) The computer readable medium of claim 28, wherein the sending step further comprises:

determining whether code related to the requested task is present on the selected server; and

downloading the code onto the selected server if the code is not present on the selected server.

39. (Previously presented) The computer readable medium of claim 28, wherein the sending step further comprises:

providing the task as a parameter to the generic compute method.

40. (Previously presented) The computer readable medium of claim 28 further comprising the step of indicating to the server that results from a computed task should be stored in a result cache on the selected server for subsequent tasks to use.

41. (Previously presented) The computer readable medium of claim 28, wherein the results are used for further processing on the client.

42. (Previously presented) The computer readable medium of claim 28, wherein the results comprise an object.

43. (Currently amended) A computer readable medium containing instructions for controlling a computer system comprising a collection of servers to perform a method for enabling a server to dynamically receive and process a task from a client computer wherein the task is in an executable programming language compatible with each of the servers, the method comprising ~~the steps of~~:

downloading a class definition after receiving a task request, wherein the class definition maps locations of information in the task request and allows the server to process the task request;

assembling parameters and data from ~~a~~ the task request into a task, using the downloaded class definition;

~~downloading any needed executable byte code;~~

invoking a generic compute method ~~on the server~~, ~~which is~~ capable of processing a plurality of types of tasks, on the server, wherein the generic compute method ~~which~~ executes the task and generates results; and

returning results to the client.

44. (Previously presented) The computer readable medium of claim 43, wherein the computer system is operatively coupled to a primary storage device, a secondary storage device, a display device, and an input/output mechanism.

45. (Previously presented) The computer readable medium of claim 43, wherein the task is developed in a programming language compatible with each of the servers.

46. (Previously presented) The computer readable medium of claim 43, wherein the task is developed using a Java programming language and environment.

47. (Previously presented) The computer readable medium of claim 43, wherein the environment includes a remote procedure call subsystem.

48. (Previously presented) The computer readable medium of claim 47, wherein the remote procedure call subsystem is the Remote Method Invocation (RMI) system.

49. (Previously presented) The computer readable medium of claim 43, wherein the assembling step further comprises:

determining if types related to the task are available on the server;

when the types are not available on the server, downloading the types onto the server from a location as indicated by the parameters provided by the client; and

executing the task based upon the data and parameters provided by the client.

50. (Previously presented) The computer readable medium of claim 49, wherein the determining step and the downloading steps are performed by a remote procedure call (RPC) subsystem.

51. (Previously presented) The computer readable medium of claim 50, wherein the determining step is performed by a Remote Method Invocation (RMI) type of remote procedure call subsystem.

52. (Previously presented) The computer readable medium of claim 43, further comprising:

storing the results from the task in a cache if a subsequent task will use the results.

53. (New) The method of claim 3, wherein the server downloads the class definition from a location indicated by a URL parameter in the task request.

54. (New) The method of claim 3, wherein the server provides the task request as a parameter to the generic compute technique.